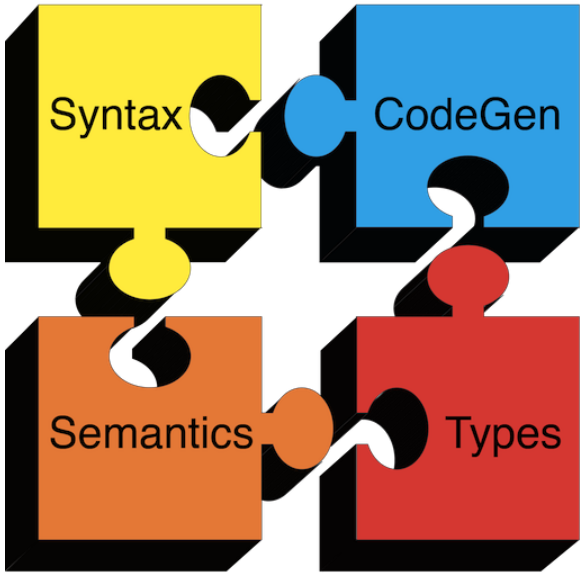


Compilers

Runtime Organization

Alex Aiken

Runtime Organization

- We have covered the front-end phases
  - Lexical analysis
  - Parsing
  - Semantic analysis
- Next are the back-end phases
  - Optimization
  - Code generation

Runtime Organization

- Before discussing code generation, we need to understand what we are trying to generate

- There are a number of standard techniques for structuring executable code that are widely used

Runtime Organization

- Management of run-time resources
- Correspondence between
  - static (compile-time) and
  - dynamic (run-time) structures
- Storage organization

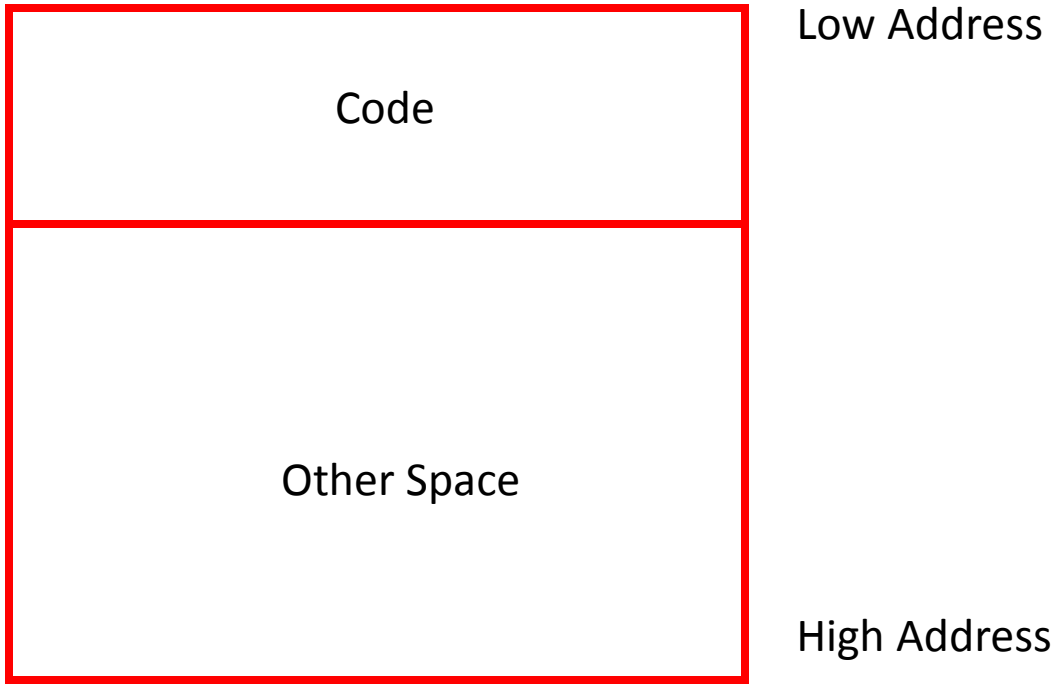
Runtime Organization

- Execution of a program is initially under the control of the operating system

- When a program is invoked:
  - The OS allocates space for the program
  - The code is loaded into part of the space
  - The OS jumps to the entry point (i.e., “main”)

Runtime Organization

Memory



Runtime Organization

- By tradition, pictures of machine organization have:
  - Low address at the top
  - High address at the bottom
  - Lines delimiting areas for different kinds of data

- These pictures are simplifications
  - E.g., not all memory need be contiguous

Runtime Organization

- Other Space = Data Space

- Compiler is responsible for:
  - Generating code
  - Orchestrating use of the data area